Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 529 – 536

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

SIGHTED: A Framework for Semantic Integration of Heterogeneous Sensor Data on the Internet of Things

Ahmad M. Nagib[a,*], Haitham S. Hamza[a]

[a]*ANSR Lab, Faculty of Computers and Information, Cairo University, 12613 Giza, Egypt*

Abstract

Sensors are embedded nowadays in a growing number of everyday life objects. Smartphones, wearables, and sensor networks together play an important role in bridging the gap between physical and cyber worlds, a fundamental aspect of the Internet of Things vision. The ability to reuse sensor data integrated from multiple heterogeneous sources is a step towards building innovative applications and services. In this paper SIGHTED, a sensor data integration framework, is proposed exploiting semantic web technologies and linked data principles. It provides a layered structure as a guideline for integrating sensor data from various sources supporting accessibility and usability. DotThing, a demo platform, is implemented as an instantiation of SIGHTED framework and evaluated. Smartphones and sensor nodes are connected to DotThing showing the ability to query and reuse integrated sensor data from multiple sources to create more flexible horizontal applications. DotThing implementation also demonstrates the need for adding a semantic layer to existing IoT cloud-based platforms, like Xively, that generally lack such layer resulting in proprietary vertical solutions with limited data integration and discovery capabilities. DotThing makes use of vocabularies from existing ontologies on the linked data cloud providing a unified model to annotate data and link it to existing resources on the web.

Peer-review under responsibility of the Conference Program Chairs

Keywords: IoT, Internet of Things, SIGHTED, data integration framework, semantic web, sensor data, linked data, Web of Things, DotThing

1. Introduction

The Internet of Things (IoT) still does not have a widely accepted definition. IoT initially considered RFID tagged physical objects. The vision grown to cover distributed sensor networks, sensor-enabled devices and generally smart objects collaborating to support services that allow interaction with physical world. IEEE IoT Initiative recently released a document to establish a common definition of IoT[1]. According to IBM Center for Applied Insights, International Data Corporation (IDC) estimated that 212 billion sensor-enabled objects will be available by 2020, 30 billion of them will be connected to networks[2]. Although there is a progress in sensor-enabled devices abstraction, applications are still immensely coupled to sensor deployments in a vertical approach. Integration of sensor data from two various systems for reuse in unintended application is still not a trivial task. This tight-coupling is a result of heterogeneous sensor data representation formats, and absence of sensor data context and sensed value meaning except for a specific proprietary application. Sensor data true potential is unexploited in such vertical perception

* Corresponding author. Tel.: +2-0100-740-2345.
E-mail address: ahmadnagib@fci-cu.edu.eg

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.04.251

of data. A vital step towards the success of the IoT vision is the reuse of data collected from widely distributed heterogeneous sensor-enabled devices. In this horizontal approach applications can use data in an integrated way regardless the technical aspects of each system. Sensors in such case represent an essential way to collect data from their environment fetching states of real-world things by generating context data. This contributes to mapping physical world's people, places and things to cyber world bridging the gap between two largely disconnected worlds. Evolution of IoT and expansion of internet-connected sensor-enabled devices has resulted in massive amounts of data raising the need for huge data storage and computing power. This resulted in the emergence of IoT cloud-based platforms like Xively[1] and OpenSensors.io[2] for storing data to be used by proprietary applications. This approach requires prior knowledge of the used devices, software development environments and platforms' libraries and APIs.

Since most of sensor data on the web depends on heterogeneous models, data needs to be made available homogeneously to allow integration from wide variety of sources. A unified machine-understandable representation of world knowledge is required to put things into common semantic context. An approach to achieve data integration from various sensor deployments is adding a semantic layer annotating sensor data according to ontology concepts preferably applying linked data principles[3]. A considerable aspect for evolving IoT is building scalable distributed web-accessible sensor data platforms. Data collection context should be in computer-interpretable fashion to facilitate discovery and access to data using standard methods. The needed mechanisms are mostly supported by semantic web technologies. Resources can be explicitly annotated with semantic metadata encoding the meaning of their capabilities and measurement values to be interpreted by machines without human intervention. This facilitates the creation of intelligent applications and supports decision making and reasoning mechanisms. Efforts are still slow-paced towards creating generic, dynamic and scalable open platforms supporting reuse of such huge data bridging data silos.

A framework for Semantic InteGration of HeTerogeneous SEnsor Data (SIGHTED) is proposed as a reference to collect and provide uniform access to heterogeneous sensor data of multiple sources based on semantic web and linked data principles. DotThing platform is instantiated based on SIGHTED layered structure to publish integrated semantically annotated sensor data to be consumed by programs. The remainder of this paper is organized as follows: Section 2 discusses background and related work. Section 3 explains proposed SIGHTED framework. Section 4 presents proposed DotThing platform and results of initial performance evaluation and Section 5 concludes the paper.

2. Background and Related Work

2.1. Internet of Things Heterogeneity and The Semantic Web

Things in the context of IoT can be divided into three main classes. The first includes devices attached to objects for identification. The second class includes sensors and actuators to provide external access to object's properties and functions. The third class is for embedded sensor-enabled devices like wearables and smartphones with embedded access to their properties and functions mostly over the internet. Regardless the way in which things are connected to IoT, they should be uniformly discoverable and integrated with network infrastructure and its existing services which requires interoperability at multiple levels. IoT smart environments are growing in numbers and varieties of connected devices vendors. Developers should be able to build applications based on heterogeneous data independent of hardware and software capabilities. Communication and networking of smart objects within the IoT have been in research focus with some efforts of standardization[4,5,6]. Less focus is given on the application level to provide integrated web-accessible data in a unified way after data is collected from such multiple heterogeneous environments.

Open distributed platforms are needed to support such integration and to provide context and enhance knowledge extraction from sensor data. The Semantic Web[7] is an extension of current web in which the meaning of information is made explicit using ontologies. It can be considered as a huge distributed knowledge base of machine-understandable data. Semantic annotation of sensor data can be described as adding semantic tags to raw data to represent concepts, properties and relationships based on an ontology to describe the metadata associated to sensor data in a meaningful standardized way. This allows programs to automatically process information meaning integrating and relating heterogeneous data. It also allows reasoning capabilities by implicitly inferring logical consequences and new facts about sensor data where ontology languages encode domain knowledge and inference rules. Semantic web extends

[1] http://xively.com/
[2] http://www.opensensors.io/

past attempts to expose things as web resources adding the advantages of using standard languages, domain models linkage, extensibility, and web scale using URIs and HTTP. This approach is needed to easily access heterogeneous sensor data on large scale and to integrate physical objects with cyber world. Resource Description Framework (RDF) describes information in a machine-understandable way where sensor data is described in terms of properties and values using RDF triples. This forms statements describing readings value, type, location, and other context information. SPARQL queries are used to access RDF data based on required sensors features or dynamic states.

The past years witnessed efforts to model sensor-related features[8] creating ontologies that cover these aspects and provide meaningful connections between data sources in addition to RDF models that guarantee extensibility. Linked data refers to available annotated data on the web identified by URIs, accessible via HTTP and linked to other resources. Similarly, physical world data can be interlinked to domain knowledge and existing data sources on the web to provide additional information and to facilitate automated annotation and reasoning. This enables better accessibility, and supports integration with existing knowledge on Linked Open Data (LOD) cloud. It also increases usability of published data and enables obtaining information from various domains relating sensor data to web resources. Linked data guarantees generic API for heterogeneous data sources enabling simple data sharing among applications via HTTP. It is good to reuse existing standard domain ontologies and upper ontologies to unify high-level concepts in various applications. These ontologies should be extended according to intended application's logic as it is insufficient to rely only on general definitions. About 9960 datasets were published and interlinked as of 2015[3] which shows huge improvement in adapting to semantic web compared to 203 datasets until 2010.

2.2. Related Work

Integrating resource-constrained devices into the internet is difficult as Internet Protocols are resource demanding. 6lowPAN[9], and CoAP[4] are lightweight alternatives that can be converted to and from Internet Protocols. Clients can use CoAP on top of UDP with 6lowPAN to query sensors. However, this only allows processing data provided by known set of sensors by manual integration for proprietary applications which does not scale well. Much efforts were directed towards providing HTTP access to sensor observations as in OGC's Sensor Web Enablement (SWE)[5]. This approach is restricted by system-specific schema for publishing sensor data. SWE's Sensor Observation Service does not support semantic interoperability or reasoning[10]. Semantic Sensor Web[11] proposed exploiting semantic technologies to annotate sensor data with metadata related to location, time and other aspects. In[12] available linked data is reused to annotate sensor features without creating redundant data. In[13] mechanisms to search for sensors in OGC SWE are proposed exploiting primitive semantic relationships but not exposing linked data. In[14] a framework is proposed by which sensor data is converted to RDF and linked to existing data on LOD cloud. A sensor ontology schema based on O&M concepts is used with links to GeoNames[4] dataset for location properties.

Trends of publishing sensor data on the LOD cloud are discussed in[15] to improve accessibility without increasing complexity of solutions. The work in[16] supports semantic sensor discovery without exploiting hierarchical and structured relations. Sense2Web[17] is a platform that captures basic attributes of sensors in RDF using available linked data to create links to other resources. Sensor descriptions are manually submitted and stored in XML format to be transformed into RDF. The platform focuses on publishing sensor features only with no attention to sensor readings. The demo implemented only retrieves and shows published sensor features in a map overlay. An ontology linked to GeoNames and QU ontology[5] is developed to model IoT resources. SEMSENSE[18] is a system for collecting and publishing sensor data of just one single source. It semantically enriches sensor data residing in MySQL database based on manual mapping to SSN ontology[19] concepts. A D2R server[20] is used to provide interface to the relational database. Concepts from wgs84_pos Basic Geo vocabulary[6], and GeoNames are used. Linked Stream Middleware (LSM)[21] is a cloud-based platform supporting sensor data collection, annotation, and publishing. LSM uses SSN ontology vocabulary and links data to existing resources on LOD cloud. It only provides a limited scenario to evaluate the query processing performance over artificially fetched historical data.

[3] http://stats.lod2.eu
[4] http://www.geonames.org/
[5] http://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu
[6] http://www.w3.org/2003/01/geo/wgs8_pos

3. The SIGHTED Framework

SIGHTED is proposed as a sensor data integration framework that exploits semantic web technologies and linked data principles. It acts as a reference to collect and provide uniform access to heterogeneous sensor data from multiple sources. It provides a layered structure that is considered as a guideline for building service platforms that aim to combine sensor data and other data from various sources on the web to be integrated and published for reuse in horizontal applications. This facilitates the creation of dynamic open platforms based on SIGHTED layered structure to acquire raw sensor data and publish interoperable semantically annotated data taking context into consideration. As shown in Fig. 1, SIGHTED structure is divided into five layers covering the entire process from collecting raw data to consuming published annotated data via software programs as described in the following subsections.

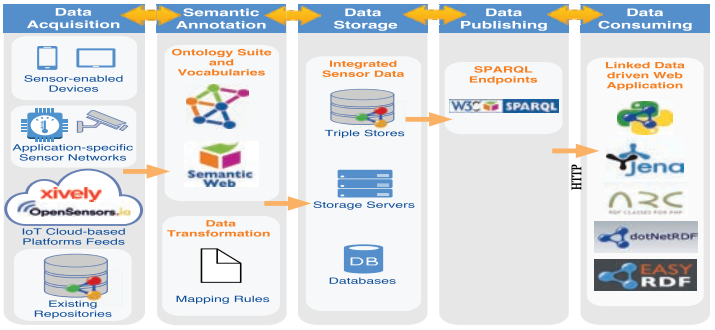

Fig. 1: SIGHTED layered structure

3.1. Data Acquisition Layer

This layer is mainly concerned with providing an interface to various IoT data sources. It is responsible for collecting data from the connected IoT devices and other resources. A system following the framework should provide wrappers to deal with large number of various input formats providing a unified format as an output. The acquired data should be transformed according to the semantic annotation layout defined in the next layer.

3.2. Semantic Annotation Layer

This layer provides a unified data model to semantically annotate captured data from various sources using a common ontology layout for data transformation into RDF. In order to develop large scale solutions, existing ontologies should be reused with no need to design a specific ontology from scratch. Sensor domain ontologies should be utilized and extended beside other relevant domain and upper ontologies. Semantic Annotation Layer provides the base for uniform access not only to sensor data collected by the Data Acquisition layer, but also to existing resources on LOD cloud according to the model used. The data model should be lightweight to reduce traffic and processing time.

3.3. Data Storage Layer

This layer is responsible for storing semantically annotated data. RDF triple stores should be used to store integrated data after being enriched using the model provided by the Semantic Annotation layer. Other kind of databases like existing relational databases should be supported providing RDF interface over them. This interface maps tables and records of the database to concepts, relations, and instances according to the Semantic Annotation layer model.

3.4. Data Publishing Layer

This layer is concerned with publishing integrated data via standard interfaces providing accessibility. SPARQL endpoints should be used to support queries on top of semantically annotated data. It facilitates access to heterogeneous data hindering their technical aspects such as software development environment and access formats.

3.5. Data Consuming Layer

In this layer data is made available for reuse by users and application developers via unified access method. The query processing facility provided by Data Publishing Layer support a simple means for rapid application development. Application code uses suitable programming libraries to query SPARQL endpoints combining data from

multiple sources in a single application program. This provides usability, and discovery of sensor data based on required sensors features or dynamic states.

4. DotThing Platform

4.1. DotThing Implementation

DotThing is a demo platform that is proposed and implemented as an instantiation of SIGHTED framework to demonstrate the functions of its various layers. DotThing integrates multiple sources' sensor data collected from smartphones and sensor nodes of heterogeneous WSNs as shown in Fig. 2. Specifications of the devices connected to the platform are shown in Table 1. It shows how complex it is to build proprietary IoT solution from scratch in terms of software and hardware diversity. DotThing intuitively hide such complexity.

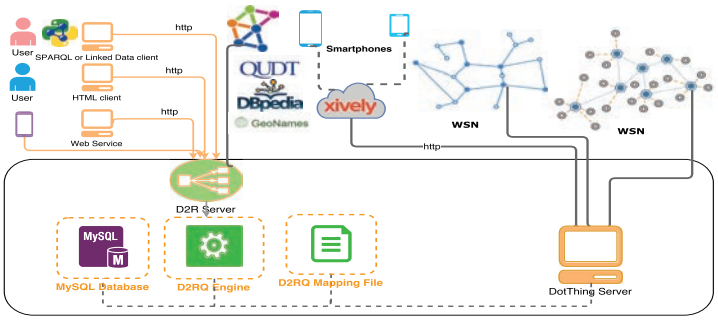

Fig. 2: DotThing platform's architecture

Xively is utilized to provide an intermediary maintainable way to collect used smartphones' sensor data. ANSRoid android application is built to get sensor readings from the used smartphones to be pushed into Xively feeds using its REST API given the feed ID and API endpoint URL. Data is then collected by DotThing server which communicates with Xively to continuously get new readings. This demonstrates DotThing's ability to deal with heterogeneous data sources. This is also investigated to show the need for semantic layer in existing IoT cloud-based platforms like Xively. Feeds' metadata on such platforms is not uniformly annotated resulting in an inefficient use of the available resources. Data feeds and channels can only be manually tagged with custom tags with no unified semantics or syntax reference. This results in limited resource discovery and data integration capabilities where DotThing platform bridges this gap.

Table 1: Specifications of DotThing's connected data sources.

Data Source	Location	Model	Sensors	Programming Language	Direct Sink
WSN	Cairo University campus	IRIS-XM2110 nodes	MDA100 sensor board (Temperature and Light)	nesC for TinyOS	MIB520 USB Gateway on Linux-based server
WSN	Cairo University campus	TelosB nodes	Temperature, Humidity and Light	nesC for TinyOS	TelsoB Gateway on Linux-based server
Smartphone	Owned by a researcher at ANSR lab	SAMSUNG GT-N7100	Atmospheric Pressure, Light and GPS	Java for Android	Xively feed via APIs and Android libraries
Smartphone	Owned by a researcher at ANSR lab	SAMSUNG GT-S7582	Atmospheric Pressure, Light and GPS	Java for Android	Xively feed via APIs and Android libraries

Sensor data resulting from the deployed WSNs is also collected by DotThing server fulfilling Data Acquisition Layer functionality. MySQL database is updated with new readings collected from all sources carrying out the functionality of Data Storage Layer in a minimal form. DotThing uses vocabularies from SSN and IoT lite[7] ontologies for sensor domain, QUDT and QU ontologies for quantities and units. DUL ontology[22], DBpedia[23], and wgs84_pos Ba-

[7] http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite

sic Geo vocabulary are used for location. This provides an IoT information model for the Semantic Annotation layer to transform data. D2RQ Platform is set up where a D2R server is provided as implementation of Data Publishing layer. The D2R server offers on the fly RDF interface to the database without replicating its content as a standalone RDF store. D2RQ mapping file is customized using proposed model vocabularies to annotate sensor data in database with relevant semantics to be published as linked data through SPARQL endpoint. URI naming convention for identifying and describing resources is customized within the file. Class maps specifies how URIs are generated for instances like devices, sensors and their readings. DotThing service runs on HP computer with Ubuntu 14.04 64-bit operating system, 4 GB of RAM and Intel Core i5 2.40 GHz CPU. D2RQ version 0.8.1 is set up and MySQL version 5.5.44 is used as an underlying RDBMS. Java run-time environment with maximum heap size of 4 GB is used for D2RQ.

4.2. DotThing Evaluation

An RDF-dump of MySQL data is carried out using D2RQ platform at different instants to estimate the number of RDF triples and the Turtle format RDF file size if the relational database was replaced by an RDF triple store that uses the model described in the mapping file. Table 2 shows the cost of having a standalone RDF triple store in terms of storage given number of sensor readings collected by DotThing at different instants compared to the implemented RDF interface to DotThing's MySQL database fulfilled by D2RQ platform.

Table 2: DotThing MySQL database vs. RDF triples size

Estimated number of RDF triples	**50 K**	**250 K**	**1 M**
Approx. number of collected sensor readings	11 K	54 K	190 K
MySQL database size on Disk	1.8 MB	5.8 MB	13.1 MB
Turtle RDF-Dump file size on disk	4.9 MB	25.5 MB	91 MB

DotThing Query Set is a set of SPARQL queries proposed to assess DotThing's integration potential based on real life scenario. The queries can be used in an interactive mobile application experience based on user's context and selections. The query set is also designed to evaluate the query processing performance of the system. It contains nine defined SPARQL queries including different query forms and functions to be mixed and used considering LSM and Berlin SPARQL Benchmark[24] queries characteristics. The query set was tested against DotThing's SPARQL endpoint to assure correct results of queries before carrying out query processing performance evaluation. The following enumeration describes queries in DotThing Query Set and then they are represented in SPARQL.

Query 1: Ask whether there are devices having sensors on board in a certain place of interest.
Query 2: Identify sensors in a certain place of interest along with their properties and units.
Query 3: Get all readings of sensors measuring a desired property in a certain place and time-period of interest.
Query 4: Get the number of sensors measuring a property of interest using given GPS coordinates.
Query 5: Get the number of sensors on board for each of the registered devices on the system.
Query 6: Retrieve the number of readings on the system for each sensing property.
Query 7: Get the average value of readings of a certain sensing property at a given place of interest.
Query 8: Get highest value of a certain property in a specific location starting from a certain time until query time.
Query 9: Retrieve the latest reading value of a specific sensor of interest.

```
PREFIX ssn: <http://www.w3.org/2005/Incubator/ssn/ssnx/ssn#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX iot-lite: <http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite#>
PREFIX DUL: <http://www.loa-cnr.it/ontologies/DUL.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Prefixes used in SPARQL queries of DotThing's Query Set

```
ASK
WHERE

    {

    ?device a ssn:sensingDevice;
        DUL:hasLocation %%.


    }
```

Query 1

```
SELECT ?sensor ?property ?propertyunit
WHERE

{
 ?sensor a ssn:Sensor;
    iot-lite:hasSensingDevice ?device;
    iot-lite:hasUnit ?propertyunit;
    iot-lite:hasQuantityKind ?property.
 ?device DUL:hasLocation %%.
}
```

Query 2

```
SELECT ?value ?time
WHERE {
    ?reading a ssn:Observation;
        ssn:observedProperty %%;
        ssn:observedBy ?s;
        ssn:hasValue ?value;
        ssn:observationResultTime ?time
        FILTER (?time < %% && ?time > %%).
    ?s iot-lite:hasSensingDevice ?device.
    ?device DUL:hasLocation  %%.}
```

Query 3

```
SELECT COUNT (DISTINCT (?sensor) )
WHERE
    { ?sensor a ssn:Sensor;
        iot-lite:hasQuantityKind %%;
        iot-lite:hasSensingDevice ?device.
      ?device geo:long "%%"^^xsd:double;
        geo:lat "%%"^^xsd:double.
    }
```

Query 4

```
SELECT ?device COUNT (?sensor)
WHERE
    {
    ?sensor a ssn:Sensor;
      iot-lite:hasSensingDevice ?device.
    }
    GROUP BY ?device
    ORDER BY ?device
```

Query 5

```
SELECT ?property COUNT(?reading)
WHERE
  {
    {?reading ssn:observedBy ?sensor.}
    {?sensor a ssn:Sensor;
      iot-lite:hasQuantityKind ?property.}
  } GROUP BY ?property
    ORDER BY DESC (COUNT(?reading))
```

Query 6

```
SELECT ?sensor AVG(?value)
WHERE
  {
   ?reading a ssn:Observation;
      ssn:observedProperty %%;
      ssn:hasValue ?value;
      ssn:observedBy ?sensor.

   ?sensor iot-lite:hasSensingDevice ?dev.
   ?dev DUL:hasLocation %%.
  }
   GROUP BY ?sensor
```

Query 7

```
SELECT ?sensor MAX (?value)
WHERE {
 ?reading a ssn:Observation
   ssn:observedProperty %%;
   ssn:hasValue ?value;
   ssn:observedBy ?sensor
   ssn:observationResultTime ?timestamp
   FILTER (?timestamp > "%%").
?sensor iot-lite:hasSensingDevice ?device.
?device DUL:hasLocation  %%. }
GROUP BY ?sensor
ORDER BY DESC (?timestamp)
```

Query 8

```
SELECT ?value ?timestamp
WHERE

  {
   ?reading a ssn:Observation;
     ssn:observedBy %%;
     ssn:hasValue ?value;
     ssn:observationResultTime ?timestamp.
  }

   ORDER BY DESC (?timestamp)
   LIMIT 1
```

Query 9

A Python program that uses SPARQL wrapper library[8] is built to query and consume DotThing's dynamically integrated data using the defined query set. System's response time is measured as an indicator of the system's performance. In this context response time is defined as the time in seconds from which a consumer program initiates a SPARQL query until the program gets back query result from DotThing's SPARQL endpoint. A test was run locally to avoid network latencies. Guided by Berlin Benchmark, the system's cache is warmed up by running the query set for 10 times using varying parameters within the queries. After the warm up process, 50 query sets are executed and the average response time per query is measured in seconds. Parameters shown as "%%" in queries are changed at run-time for each run to simulate real usage scenario. This is repeated for varying storage instants of DotThing with estimated 50K, 250K, and 1M RDF triples. The service is restarted before each evaluation run for each storage case.

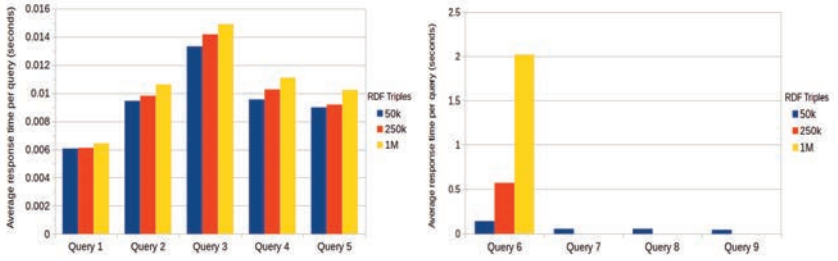

Fig. 3: Average response time per query for different storage sizes

The queries in DotThing Query Set are classified into two main classes according to their complexity and this is reflected in the resultant average response time as shown in Fig.3. The delay for delivering query results to the consumer program by DotThing's SPARQL endpoint increases for all the queries in DotThing's Query Set with the increase in storage size. The response time increases significantly for Query 6 in case of 1 M triples compared to the first five queries. This poses an undesirable restriction on the number of stored readings for performing more complex queries especially those that includes SPARQL aggregate functions such as AVG and MAX. It should be noted that queries 7 and 8 were not included in the evaluation runs of the last two storage cases and Query 9 was not included in the evaluation run of the last storage case as they all exceeded a custom defined threshold time in those cases. System response time to these queries didn't converge even after completing all of the warm up runs. While Query 6 had relatively higher response time (about double) than that for Queries 7, 8, and 9 in case of 50K triples, it is interesting to observe that DotThing server still succeeded to respond to Query 6 in the three storage cases but didn't succeed to respond in an adequate time for the other three queries.

[8] http://rdflib.github.io/sparqlwrapper/

5. Conclusion

The proposed SIGHTED framework shows the potential of the availability of sensor data driven from multiple heterogeneous sources for reuse away from building proprietary applications. DotThing platform is implemented based on SIGHTED reference framework to facilitate on the fly dynamic integration, discovery and access to various sensor data. DotThing guarantees horizontal decoupled use of senor data for building applications and services unlike vertical devices-applications coupled approach in platforms lacking semantic support like Xively. DotThing Query Set is proposed to assess DotThing's integration potential based on real sensor data dynamically collected by the platform unlike LSM that only provides a limited scenario over artificially fetched historical data. DotThing platform extends SEMSENSE to deal practically with multiple heterogeneous sensor data sources compared with single source in SEMSENSE case that also lacks a demo for data consumption and performance evaluation. DotThing adds semantics to both sensor descriptions and observations unlike Sense2Web that only focuses on annotating sensor descriptions. The rationale for current DotThing components setup is to demonstrate the functions of SIGHTED proposed layers with planned extension to compare the platform's performance using different components at each layer of SIGHTED. This will allow the investigation of the most adequate components for various scenarios of engaged sources, scale and so on. Scalability of the platform should be enhanced as the results of query processing performance evaluation show undesirable restriction on the number of stored readings specially with the more complex queries. Large scale distributed repositories and aggregation of a set of raw data through a period of time should be taken into consideration as both would extensively enhance the scalability of the platform in the context of expected voluminous IoT data.

References

1. R. Minerva, A. Biru, D. Rotondi, Towards a Definition of the Internet of Things (IoT), `http://iot.ieee.org/definition.html`, [Online; accessed 30-September-2015] (2015).
2. B. Chamberlin, The next phase of the Internet: The Internet of Things, `http://ibmcai.com/2014/06/25/the-next-phase-of-the-internet-the-internet-of-things/`, [Online; accessed 30-September-2015] (2014).
3. C. Bizer, T. Heath, T. Berners-Lee, Linked data-the story so far, Semantic Services, Interoperability and Web Applications: Emerging Concepts (2009) 205–227.
4. Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (coap).
5. M. Botts, G. Percivall, C. Reed, J. Davidson, Ogc® sensor web enablement: Overview and high level architecture, in: GeoSensor networks, Springer, 2008, pp. 175–190.
6. T. Gu, H. Pung, D. Zhang, Toward an osgi-based infrastructure for context-aware applications, Pervasive Computing, IEEE 3 (4) (2004) 66–74. `doi:10.1109/MPRV.2004.19`.
7. T. Berners-Lee, J. Hendler, O. Lassila, et al., The semantic web, Scientific american 284 (5) (2001) 28–37.
8. M. Compton, C. A. Henson, L. Lefort, H. Neuhaus, A. P. Sheth, A survey of the semantic specification of sensors.
9. Z. Shelby, C. Bormann, 6LoWPAN: The wireless embedded Internet, Vol. 43, John Wiley & Sons, 2011.
10. H. Abangar, P. Barnaghi, K. Moessner, A. Nnaemego, K. Balaskandan, R. Tafazolli, A service oriented middleware architecture for wireless sensor networks, in: Proceedings of future network and mobile summit conference, 2010.
11. A. Sheth, C. Henson, S. S. Sahoo, Semantic sensor web, Internet Computing, IEEE 12 (4) (2008) 78–83.
12. W. Wei, P. Barnaghi, Semantic annotation and reasoning for sensor data, in: Smart Sensing and Context, Springer, 2009, pp. 66–76.
13. S. Jirka, A. Bröring, C. Stasch, Discovery mechanisms for the sensor web, Sensors 9 (4) (2009) 2661–2681.
14. H. Patni, C. Henson, A. Sheth, Linked sensor data, in: Collaborative Technologies and Systems (CTS), 2010 International Symposium on, IEEE, 2010, pp. 362–370.
15. C. Keßler, K. Janowicz, Linking sensor data-why, to what, and how?, in: SSN, 2010.
16. J. Pschorr, C. A. Henson, H. K. Patni, A. P. Sheth, Sensor discovery on linked data.
17. S. De, T. Elsaleh, P. Barnaghi, S. Meissner, An internet of things platform for real-world and digital objects, Scalable Computing: Practice and Experience 13 (1).
18. A. Moraru, D. Mladenic, M. Vucnik, M. Porcius, C. Fortuna, M. Mohorcic, Exposing real world information for the web of things, in: Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011, ACM, 2011, p. 6.
19. M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al., The ssn ontology of the w3c semantic sensor network incubator group, Web Semantics: Science, Services and Agents on the World Wide Web 17 (2012) 25–32.
20. C. Bizer, R. Cyganiak, D2r server-publishing relational databases on the semantic web, in: Poster at the 5th International Semantic Web Conference, 2006, pp. 294–309.
21. D. Le-Phuoc, H. Q. Nguyen-Mau, J. X. Parreira, M. Hauswirth, A middleware framework for scalable management of linked streams, Web Semantics: Science, Services and Agents on the World Wide Web 16 (2012) 42–51.
22. A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, Sweetening ontologies with dolce, in: Knowledge engineering and knowledge management: Ontologies and the semantic Web, Springer, 2002, pp. 166–181.
23. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, Springer, 2007.
24. C. Bizer, A. Schultz, Benchmarking the performance of storage systems that expose sparql endpoints, World Wide Web Internet And Web Information Systems.